Improved plane-sweep based algorithm for boundary extraction of 3D images

Dolors Ayala Enric Vergara
Divisió Informàtica Gràfica, CREBEC
Universitat Politècnica de Catalunya
08028 Barcelona, Espanya
dolors.ayala@upc.edu evergara@lsi.upc.edu

Abstract

There exist several approaches to extract the boundary of a 3D image. Most of them represent the extracted boundary as a collection of a large number of little triangular or quadrangular faces whereas few approaches give general orthogonal faces (with any number of edges and with possible holes). One of these approaches is based on a secondary model EVM and focusses mainly on the process to obtain the orientation of the output primitives (edges and faces). Actually, this algorithm obtains, for each plane, a set of oriented edges that have to be rearranged as contours and these contours have to be classified in order to have the corresponding inclusion relationships. These last two processes are performed in a simple brute force way. In this paper, we present an improved algorithm that processes all the edges of a plane and, following a plane-sweep based method, obtains the contours and the inclusion relationships.

1 Introduction and related work

Boundary extraction of a 3D image (or otherwise said isosurface extraction of a volume dataset) is approached by two main methods, polygonal and digital. Polygonal (or beveled-form) methods represent the surface as a set of polygons (mostly triangular) [10]. Digital (or block-form) methods represent the surface as a set of voxels or a set of voxel boundary faces (surfels) [3], [16], [18]. These faces can also be general orthogonal faces bounded by four or more edges, not necessarily convex and with possible holes [2].

Digital models exhibit formal properties such as closure, orientedness and connectedness whereas polygonal models and related techniques are devoted mainly to visualization purposes and still lack of a solid modeling foundation [6].

A common drawback of the existing techniques is that the resulting isosurfaces consist of lots of little quadrangular or triangular faces. Several adaptive attempts have been developed to reduce this redundancy [13], [11], [17] but most of them actually make a post-process and produce cracks.

In [12] the authors present an approach to compute the boundary of an octree. They use a secondary PCS representation which, in the 2D case, is a list of stripes and compute the boundary as a list of polygon borders (2D) or a list of faces (3D). The 2D algorithm obtains oriented contours but it doesn't mention how obtains the inclusion relationships between them.

The problem of inclusion relationships between contours has been addressed by several authors using different denominations. Gargantini [4] uses the term of region containment tree for the same concept extended to 3D and presents a method to obtain it from an octree representation of a volume data. In [5] the authors compute the zonal graph and use it to perform several 2D image processes. Park et al. [14] present an algorithm that extracts the boundary of a 2D image. They work in the filed of NC-data generation and apply their algorithm to obtain a B-Rep of the cutting areas. They use a run-length initial model and devise an algorithm which is $O(n)$, n being the number of runs, which computes all the contours and the inclusion relationships. Although they obtain faces and holes as general orthogonal

contours, the vertical edges of them are represented by little segments of pixel size which is a consequence of the initial RL model used (see Figure 1).

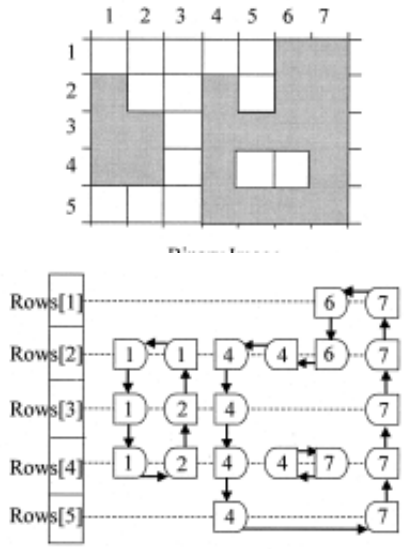


Figure 1: Example of the method using run-length (from [14]). Binary image (top) and contours obtained (bottom)

Space-sweep and plane-sweep techniques [15] are well-known and extensively used techniques in a lot of computational geometry problems as point location and convex hulls, among others.

The *Extreme Vertices Model* (*EVM*) is a very concise model whose domain are orthogonal polyhedra and, therefore, it can represent binary 3D images or volumes. It can represent manifold, i.e. well-composed [9], as well as non-manifold topologies. The EVM is actually a complete (unambiguous) solid model [1] and a complete B-Rep model can be obtained from the EVM [2] with faces and holes as general orthogonal contours which are not restricted to any size as are in the above mentioned approach of Park et al. [14]. Actually, the EVM to B-Rep conversion algorithm obtains, for each plane, a set of oriented edges that have to be rearranged as contours and these contours have to be classified in order to have the corresponding inclusion relationships. These two last processes are performed in a simple brute force way.

In this paper, we present an algorithm which solves the 2D problem of obtaining the contours and inclusion relationships from a set of edges on a plane. This algorithm can substitute the mentioned two last processes of the previous algorithm and follows a plane-sweep based strategy which takes profit of the model EVM characteristics.

The paper is arranged as follows. Next section 2 introduces the EVM and explains the previous EVM to B-Rep conversion method. Section 3 explains the new algorithm and section 4 discusses the obtained results. Finally, section 5 concludes this paper and points out future work.

2 Boundary extraction using the EVM representation

In this section, we include a short review of the EVM and the existing boundary extraction method, to make the paper more self-contained.

Let P be a 3D orthogonal polyhedron (OP). A *brink* is the maximal uninterrupted segment built out of a sequence of collinear and contiguous two-manifold edges of P. The ending vertices of a brink are called *extreme vertices* (*ev*). The *EVM* represents OP by its (and only its) set of *ev* and can recursively represent 2D and 1D orthogonal objects.

A *cut* (C) is the set of vertices of P lying on a plane perpendicular to a main axis of P. A *slice* is the region between two consecutive cuts. A *section* (S) is the resulting polygon from the intersection between P and an orthogonal plane. Each slice has its representing section. See Figure 2. Sections can be computed from cuts and vice-versa:

$$\overline{S_i(P)} = \overline{S_{i-1}(P)} \otimes^* \overline{C_i(P)}, i = 1 \ldots n,$$

$$S_0(P) = S_n(P) = \emptyset$$

$$\overline{C_i(P)} = \overline{S_{i-1}(P)} \otimes^* \overline{S_i(P)}, i = 1 \ldots n$$

where $\overline{C_i(P)}$ and $\overline{S_i(P)}$ denote the projections of $C_i(P)$ and $S_i(P)$ onto a main plane parallel to P, n is the number of cuts and $\otimes^*$ denotes the regularized XOR operation. Applying the definition of the $\otimes$ operation, this last equation can be expressed as:

$$\overline{C_i(P)} = \overline{S_{i-1}(P)} \otimes^* \overline{S_i(P)} =$$

$$(\overline{S_{i-1}(P)} -^* \overline{S_i(P)}) \cup (\overline{S_i(P)} -^* \overline{S_{i-1}(P)})$$

and, thus, any cut, C, is decomposed into two terms named *forward difference*

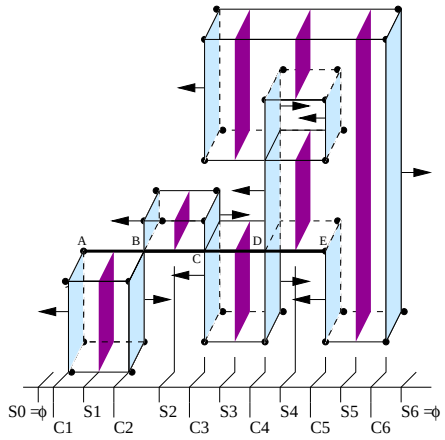


Figure 2: An OP with a marked brink from vertex A to vertex E and with non-manifold edges and vertices. Cuts and sections for one direction are shadowed and FD and BD for each cut are remarked with arrows.

$(\overline{FD_i(P)} = (\overline{S_{i-1}(P)} -^* \overline{S_i(P)})$ and *backward difference* $(\overline{BD_i(P)} = (\overline{S_i(P)} -^* \overline{S_{i-1}(P)}))$.

$FD_i(P)$ is the set of faces on $C_i(P)$ whose normal vector points to one side of the main axis perpendicular to $C_i(P)$, while $BD_i(P)$ is the set of faces pointing to the opposite side. In the example of Figure 2, FD are marked with right-pointing arrows and BD with left-pointing arrows .

This property guarantees correct orientation and, together with the fact that non-extreme vertices can be obtained from *ev*, provide proof that the EVM is a complete solid model. See [1] for more explanations concerning EVM.

Operations on the EVM, as section and Boolean operations exploit the fact that the set of *ev* can be ordered following an spatial criterion and apply space and plane-sweep based techniques [15].

In [2] and [7] an algorithm that extracts the boundary from a given volume dataset based on the EVM is presented. This algorithm converts first the voxel model to the EVM and then extracts the surface (a complete B-Rep model) that consists of general orthogonal faces.

The EVM to B-Rep conversion algorithm works recursively in the dimension. It performs three sweeping processes in order to obtain the three sets of faces parallel to the main planes. From C it computes sections and from them, FD and BD. In this process the non-extreme vertices are also obtained. The same process is performed in 2D for each C, recursively. The output of this algorithm is a set of oriented edges, for each oriented plane.

Then, for each oriented plane, contour construction and inclusion relationships for the corresponding set of edges are performed as a post-process following two simple brute-force strategies: a domino-like procedure and exhaustive point inclusion tests, respectively.

3 New 2D EVM to B-Rep conversion algorithm

Our proposal is focussed on the 2D EVM to B-Rep conversion. It presupposes that the previous 3D EVM to B-Rep conversion algorithm has been executed and that we have, for each C, its FD and BD with their corresponding normal vectors. Note that any FD and BD is a 2D EVM composed by 1D cuts (cuts from now on). Therefore, we have a set of 2D EVM to process. For instance, in Figure 2 C_4 has a FD with 2 contours (2 cuts with 2 brinks each) and a BD with 1 contour (2 cuts with 1 brink each).

The input of the presented method is a 2D EVM and the output is the complete set of oriented contours (faces and holes) and their inclusion relationships. See Figure 3.

The main difference between this new approach and the approach explained in the previous section is that in this new approach all the processes involved exploit the vertices ordering and apply plane-sweeping based techniques as most of the methods using EVM.

The method performs two plane-sweeping processes only in one direction, say X. The first one computes the 1D sections and the 2D contours, as sets of vertical brinks, while the second one classifies the contours as face or hole and obtains the inclusion relationships between them. The 1D sections are stored because the second process needs them.

In 2D, the concept of non-extreme vertex corresponds with that of non-manifold and present the topology shown in Figure 4 (this is not true in higher dimensions). These vertices aren't explicitly represented in the EVM, but can be computed from it and this computation is performed in the first process.

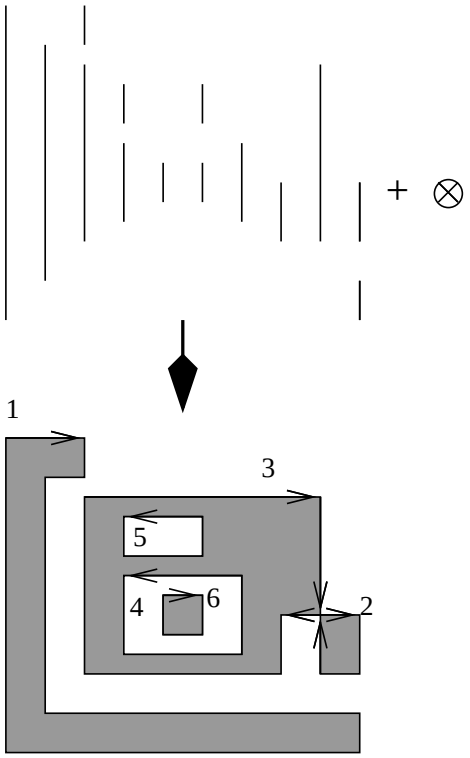


Figure 3: A 2D EVM with 10 cuts (above) corresponding to a FD or a BD plus the normal vector associated ($\otimes$ stands for a vector pointing inside) and the complete boundary obtained (below): faces 1, 2 and 6 without holes and face 3 with holes 4 and 5. Faces 2 and 3 share a non-manifold (non-extreme) vertex

The first sweeping process maintains two lists. One list with the *ev* of the active section in which each *ev* has a pointer to the contour to which belongs and another list of contours, each with the brinks already classified as belonging to this contour. The process is summarized as follows. Each new cut C is merged with the active section computing the next section and doing at the same time a contour classification.

Merging a section and a cut is an XOR process between vertices of both sets (actually the XOR is performed with the projection of the entities onto a main parallel line). If a vertex of the cut matches with one of the active section, it implies that the vertex of the cut belongs to the contour associated

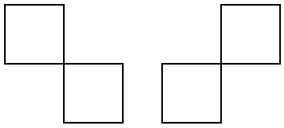


Figure 4: Non-manifold configurations corresponding also to non-extreme vertices.

with the vertex in the active section (two vertices match if their projections to the mentioned main parallel line coincide). Then we can classify each brink (ev1, ev2) of the cut as follows:

- if (ev1, ev2) do not belong to any contour, then add a new contour with this brink as an edge
- if (ev1, ev2) belongs to the same contour, then add this brink as a new edge of the contour
- if (ev1, ev2) belongs to two different contours, then merge both contours in one and add this brink as a new edge

To sum up, this first process performs a partition of the vertical brinks, each equivalence class of it being the set of vertical edges of a contour.

In this first process, when a non-manifold vertex is encountered, it has to be labeled and, at the end, a traversal of all the non-manifold vertices is needed to classify those brinks that have one such ending vertex. A non-manifold vertex is detected when a brink of the current cut contains one *ev* of the current section (always in the projections to the mentioned main parallel line).

The second pass of the method consists also in a plane-sweeping process and labels the contours as face or hole and obtains the inclusion relationships between them. It simply traverses the sections computed in the first process looking up to the contour associated with each vertex of the current section. These lists of contours reflect in 1D the 2D situation and, therefore, the inclusion tests needed to classify the contours are actually made in 1D. Note that, in this sweeping process, the contours corresponding to faces will appear before to those of their corresponding holes and that all the contours appearing in the first section correspond to faces. Then, when processing a section, we can classify as face or hole all the contours not yet classified by testing 1D inclusion relationships. So, a contour C_H is a hole of

a contour C_F if $C_H \subset C_F$ and C_F is the most internal face contour that matches this condition in the current section. The classification of a contour as a new face follows the same strategy.

Both sweeping processes have to be performed only in one direction, say X. Finally, the contours, as sets of vertical edges, already classified as faces or holes are ordered clockwise or counterclockwise depending on the normal vector direction of the 2D EVM processed.

In order to clarify these two processes, we include a working example (see Figure 5). For the first step we use the following notation: E is the list of extreme vertices (ev) with their associate contour (c) of the current section, $E = [(ev_{i1}, c_{i1})(ev_{f1}, c_{f1})] \ldots [(ev_{in}, c_{in})(ev_{fn}, c_{fn})]$, n being the number of brinks of the section and i and f standing for the initial and final ev of the brink; L is the current list of contours with their corresponding lists of brinks which are already classified as edges, $L = [c_1, \{(ev^1_{1i}, (ev^1_{1f}), \ldots\}, \ldots, c_m, \{(ev^m_{1i}, (ev^m_{1f}), \ldots]$, m being the total number of contours that will be found. We show the process for 6 cuts.

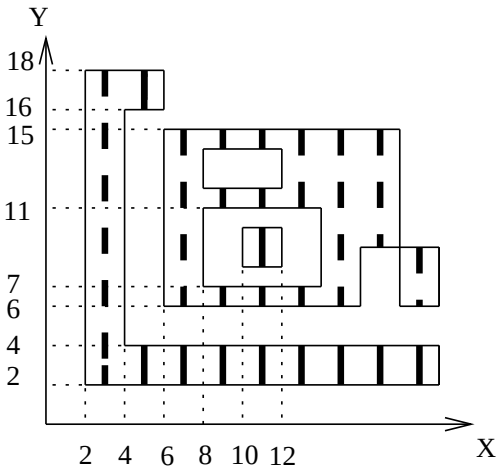


Figure 5: Working example. Thick lines show the sections.

C_1: brink $(2,18)$
$S_1 = [(1,1)(18,1)]$, $L = [1\{(1,18)\}]$

C_2: brink $(4,16)$
$S_2 = [(2,1)(4,2)(16,2)(18,1)]$
$L = [1\{(1,18)\}, 2\{(4,16)\}]$

C_3: brinks $(6,15),(16,18)$. The second brink allows to merge contours 1 and 2.
$S_3 = [(2,1)(4,1)(6,3)(15,3)]$
$L = [1\{(1,18),(16,18),(4,16)\}, 3\{(6,15)\}]$

C_4: brinks $(7,11)(12,14)$
$S_4 = [(2,1)(4,1)(6,3)(7,4)(11,4)(12,5)(14,5)$
$(15,3)]$
$L = [1\{(1,18),(16,18),(4,16)\}, 3\{(6,15)\},$
$4\{(7,11)\}, 5\{(12,14)\}]$

C_5: brink $(8,10)$
$S_5 = [(2,1)(4,1)(6,3)(7,4)(8,6)(10,6)(11,4)$
$(12,5)(14,5)(15,3)]$
$L = [1\{(1,18),(16,18),(4,16)\}, 3\{(6,15)\},$
$4\{(7,11)\}, 5\{(12,14)\}, 6\{(8,10)\}]$

C_6: brinks $(8,10)(12,14)$
$S_6 = [(2,1)(4,1)(6,3)(7,4)(11,4)(15,3)]$
$L = [1\{(1,18),(16,18),(4,16)\}, 3\{(6,15)\},$
$4\{(7,11)\}, 5\{(12,14)_{x=8}(12,14)_{x=12}\},$
$6\{(8,10)_{x=10}(8,10)_{x=12}\}]$

The process continues until the 10th cut. The 9th cut has a non-manifold vertex that will be labeled treated at the end separately. For the second process, the computed and stored sections are used to obtain the inclusion relationships. S_1 is used to label c_1 as a face. S_2 doesn't contribute because there is any new contour to classify (c_2 has been merged with c_1). S_3 is used to label c_3 as a face. S_4 is used to label c_4 and c_5 as holes of face c_3. S_5 is used to label c_6 as a new face and in S_6 there aren't new contours to classify.

The example shown in Figure 6 shows better this second process. The current section $S_c = [(,1)(,2)(,3)(,3)(,4)(,4)(,2)(,5)(,5)(,1)(,6)(,6)]$ has the contours 1 and 2 already classified as face and hole respectively and allows to classify contours 3 and 4 as new faces (they are included in contour 2 which is a hole), contour 5 as a hole of contour 1 and contour 6 as a new face.

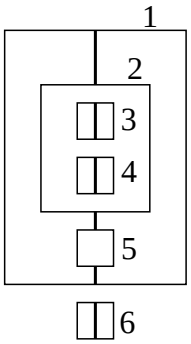


Figure 6: Working example for the inclusion relationships process

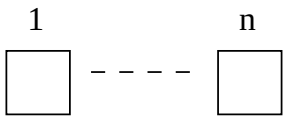
4 Results

In this section we will compare the previous brute-force method (BF) and the new approach presented in this paper (NA). Doing a theoretic complexity analysis of both methods is beyond the scope of this work. So, we first will evaluate the complexity of both methods performing a qualitative analysis of the involved processes and then we will show running times for several representative phantom examples and for two real cases.

The BF method performs a plane-sweep and computes sections (XOR operation) and 1D FD and BD (1D Boolean difference operations). This gives the full set of edges oriented of this plane. The same process has to be performed for both directions, X and Y. Then, the contours are computed performing a domino-like process among all the edges and the contours are labeled as face or hole depending on the orientation of their edges. Finally, for each hole, the method performs a search among the faces to find to which face it belongs.

The NA method also computes sections but only in one direction and at the same time maintains and computes the list of contours. Then it needs to traverse again the sections (but not to compute them again) to obtain the inclusion relationships between contours and without any point-in-polygon inclusion test. Finally, it needs to perform domino-like processes, but among half of the edges (the vertical ones) of each contour separately.

We present the following phantom examples. All the running times are in seconds and the computer used to do these comparisons is a Pentium III at 997 MHz and 512 RAM. The first one is a sequence of simple parallelepipeds. Figure 7 shows the schematic example and running times. We can see that the behavior of the NA method is linear while the BF is almost quadratic. We have performed a similar test, but with a sequence of objects as those of Figure 4 (with a non-manifold vertex each), and the running times are slightly superior but follow the same pattern.



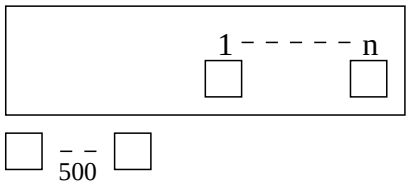
n	BF	NA
10	0.01	0.03
100	0.23	0.09
200	0.55	0.15
300	1.23	0.21
400	1.79	0.25
500	2.60	0.29

Figure 7: First example with running times

The second and third tests are intended to evaluate cases with several holes and with several levels of inclusion respectively. Figures 8 and 9 show the schematic examples and running times. Both cases show that the running times in both methods are greater than in the simple case showed, but that the new approach performs rather better than the previous one.

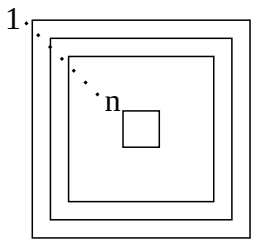
Finally we have processed two real examples. We have applied both methods to all the planes of vertices in one direction and we have computed the average times. The first example corresponds to a skull shown in Figure 10 and has 123 X-cuts, 176 Y-cuts and 129 Z-cuts and the average running times are BF = 0.056 sec. and NA = 0.015 sec. The second example corresponds to the well-known engine dataset shown in Figure 11. It has 276 X-cuts, 389 Y-cuts and 213 Z-cuts and the average running times are BF = 0.112 sec. and NA = 0.026 sec.

The time of the BF in both cases is about 4 times the time of the NA. As we have expected, these real cases don't give the time differences of several phantom examples, which have been analyzed



n	BF	NA
50	4.36	0.37
100	6.65	0.42
150	8.80	0.48

Figure 8: Example testing holes



n	BF	NA
10	0.06	0.03
40	0.38	0.16
80	1.45	0.46
160	5.66	1.78

Figure 9: Example testing inclusion levels

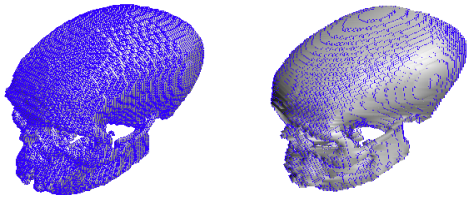


Figure 10: Skull example: voxelization (left) and 3D model EVM (right)

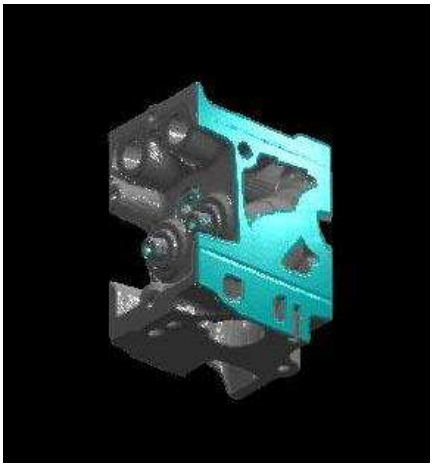


Figure 11: Engine example

to evaluate the behavior of both methods in extreme situations.

5 Conclusions and future work

An improved 2D EVM to B-Rep method has been presented that complements a 3D existing method. It basically substitutes its last process consisting of the 2D contours arrangement and classification, which is performed in a brute force way, by a plane-sweep like process which follows the same style of the main 3D algorithm and of most of the EVM involved methods.

A qualitative analysis has been performed that allow us to conclude that the new version is faster than the previous one. This conclusion has been corroborated by experimental results.

We have thought to extend this algorithm to 3D to perform connected component labeling and compute the containment tree as in [4], but there exists an algorithm that solves this problem based on the EVM [7]. Although the method is slightly different, it also follows a space-based sweeping strategy and uses the information associated to the sections of the model.

Initially the EVM was developed to represented binary datasets. In [8], it has been extended to represent general volumes and the extended model has been called VolumeEVM. As a future work, we are studying the suitability of this model to represent time-varying data sets by taking profit of spatial and

temporal coherence.

6 Acknowledgments

This work has been partially supported by the projects MAT2002-04297-C03-02 and IM3: Imagen Molecular y Multimodalidad from the Spanish government and by the CREBEC from the Catalan government.

References

[1] A. Aguilera. *Orthogonal Polyhedra: Study and Application*. PhD thesis, LSI-Universitat Politècnica de Catalunya, 1998.

[2] A. Aguilera and D. Ayala. Converting Orthogonal Polyhedra from Extreme Vertices Model to B-Rep and to Alternative Sum of Volumes. *Computing Suppl. Springer-Verlag*, 14:1 – 28, 2001.

[3] E. Artzy, G. Frieder, and H. Gabor. The theory, design, implementation and evaluation of a three-dimensional surface detection algorithm. *Computer Graphics and Image Processing*, 15:1 – 24, 1981.

[4] I. Gargantini, G. Schrack, and A. kwok. Reconstructing multishell solids from voxel-based contours. *Computer-Aided Design*, 26(4):293 – 301, 1994.

[5] H. J. A. M. Heijmans. Connected morphological operators for binary images. *Computer Visuin and Image Understanding*, 73(1):99–120, 1999.

[6] G. J.Grevera, J. K. Udupa, and D. Odhner. An Order of Magnitude Faster Isosurface Rendering in Software on a PC than Using Dedicated, General Purpose Rendering Hardware. *IEEE Trans. Vis. and Comp. Graph.*, 6(4):335 – 345, 2000.

[7] J.Rodríguez, D. Ayala, and A. Aguilera. *Geometric Modeling for Scientific Visualization*, chapter EVM: A Complete Solid Model for Surface Rendering, pages 259 – 274. Springer, 2004.

[8] J.Rodríguez, D. Ayala, and S. Grau. Volumeevm: A new approach for surface/volume integration. *Computers & Graphics*, 29:217 – 224, 2005.

[9] L. Latecki. 3D Well-Composed Pictures. *Graphical Models and Image Processing*, 59(3):164 – 172, 1997.

[10] W. Lorensen and H. Cline. Marching cubes: A high resolution 3D surfaces construction algorithm. *Computer Graphics*, 21(4):163 – 169, 1987.

[11] C. Montani, R. Scateni, and R. Scopigno. Discretized marching cubes. In *Proceedings of visualization*, pages 281 – 287, 1994.

[12] C. Montani and R. Scopigno. *Graphics Gems II*, chapter IV.7 Quadtree/Octree to boundary conversion, pages 202 – 218. Academic Press, Inc, 1991.

[13] H. Muller and M. Stark. Adaptive generation of surfaces in volume data. *Visual computer*, 9:182 – 199, 1995.

[14] S.C. Park and B. K. Choi. Boundary extraction for cutting area detection. *Computer-Aided Design*, 33:571 – 579, 2001.

[15] F. P. Preparata and M. I. Shamos. *Computational Geometry: an Introduction*. Springer-Verlag, 1985.

[16] A. Rosenfeld, T. Kong, and A. Wu. Digital surfaces. *CVGIP: Graphical Models and Image Processing*, 53(4):305 – 312, 1991.

[17] R. Shekhar, E. Fayyad, R. Yagel, and J.F. Cornhill. Octree-based decimation of marching cubes. *IEEE Computer Graphics and Applications*, pages 335 – 342, 1996.

[18] J. Udupa and O. Odhner. Fast visualization, manipulation and analysis of binary volumetric objects. *IEEE Comp. Graph. and Appl.*, 4(1):53 – 62, 1991.